

```
PLEASE ENTER HOST PORT ID:
PLEASE ENTER HOST PORT ID:x
LOGINID:d232mbg
PASSWORD:
TERMINAL (ENTER 1, 2, 3, 4, OR ?):03
```

FILE 'USPAT' ENTERED AT 08:37:47 ON 22 OCT 1998

```

* * * * *
*           W E L C O M E   T O   T H E
*           U . S .   P A T E N T   T E X T   F I L E
* * * * *

```

=> s 5768510/pn

L1 1 5768510/PN

=> s l1 and configur?

738320 CONFIGUR?
L2 1 L1 AND CONFIGUR?

=> s l1 and (configur? or custom?)

738320 CONFIGUR?
168172 CUSTOM?
L3 1 L1 AND (CONFIGUR? OR CUSTOM?)

=> d kwic

US PAT NO: 5,768,510 [IMAGE AVAILABLE] L3: 1 of 1

ABSTRACT:

An . . . computer and a network for connecting the client computer to the server computer which utilize an execution framework code segment **configured** to couple the server computer and the client computer via the network, by a plurality of client computer code segments. . .

SUMMARY:

BSUM(14)

The prior art **configuration** shown in FIG. 1 generally works well in a system where a single application program 102 is running at any. . . write information into any area of the entire screen buffer area 110 without causing a display problem. However, if the **configuration** shown in FIG. 1 is used in a computer system where more than one application program 102 can be operational. . .

SUMMARY:

BSUM(30)

A . . . that allows a user to create manageable applications, that can be readily deployed, installed on a variety of platforms, and **configured** to facilitate partitioning them on clients versus servers and administer the applications once they're running. Systems don't always break because. . .

SUMMARY:

BSUM(31)

The . . . to come in, help it develop an application. Their goal is the end application, which must be maintained. The company **configures** it, evolves it, and grows it. To allow for modification, the development task must be modular to allow different groups. . .

SUMMARY:

BSUM(35)

A . . . computer and a network for connecting the client computer to the server computer which utilize an execution framework code segment **configured** to couple the server computer and the client computer via the network, by a plurality of client computer code segments. . . .

DRAWING DESC:

DRWD(5)

FIG. 3 is a schematic block diagram of a typical hardware **configuration** of a computer in accordance with the subject invention;

DETDESC:

DETD(2)

The . . . IBM, PS/2, or Apple, Macintosh, computer. A representative hardware environment is depicted in FIG. 3, which illustrates a typical hardware **configuration** of a computer 300 in accordance with the subject invention. The computer 300 is controlled by a central processing unit. . . .

DETDESC:

DETD(3)

Specifically, . . . memory (RAM) 306 for temporary storage of information, a read only memory (ROM) 304 for permanent storage of the computer's **configuration** and basic operating commands and an input/output (I/O) adapter 310 for connecting peripheral devices such as a disk unit 313. . . .

DETDESC:

DETD(9)

The . . . a new subclass which has some of the functionality (with selective modification) of another class allows software developers to easily **customize** existing code to meet their particular needs.

DETDESC:

DETD(11)

For . . . can be inherited by developer-defined subclasses and either modified or overridden to allow developers to extend the framework and create **customized** solutions in a particular area of expertise. This object-oriented approach provides a major advantage over traditional programming since the programmer. . . .

DETDESC:

DETD(14)

In . . . a preferred embodiment, can provide a prefabricated functionality for system level services which developers can modify or override to create **customized** solutions, thereby avoiding the awkward procedural calls necessary with the prior art application frameworks programs. For example, consider a display. . . .

DETDESC:

DETD(47)

A . . . approach allows the creation of more complex systems that

work together in interesting ways, as opposed to isolated programs, having **custom** code, be created over and over again for similar problems.

DETDESC:

DETD(65)

With Java, developers can create robust User Interface (UI) components. **Custom** "widgets" (e.g. real-time stock tickers, animated icons, etc.) can be created, and client-side performance is improved. Unlike HTML, Java supports the notion of client-side validation, offloading appropriate processing onto the client for improved performance. Using the above-mentioned **custom** UI components dynamic, real-time Web pages can also be created.

DETDESC:

DETD(68)

A . . . provides a system for building manageable applications. The applications can be readily deployed, on to a variety of platforms, and **configured** so that it's easy to partition them on to clients versus servers and administer the applications. A preferred embodiment is. . .

DETDESC:

DETD(69)

Enterprise . . . program or programs for accessing the information on the server. The complexity of these environments makes it difficult to create, **configure**, deploy and administer software applications. However, the advent of Web technologies (browsers, the Java language and HTTP) has enabled enterprises. . .

DETDESC:

DETD(85)

During . . . library 720 manages the connection to facilitate policies that maximize access to its resources. For example, a server can be **configured** for a maximum time to await a transaction response. A timer runs, if it exceeds the maximum time before new. . .

DETDESC:

DETD(95)

The . . . Java startup applet enables clients to access the application on the server. A set of tools for application installation and **configuration** are provided as applications on server nodes in accordance with a preferred embodiment. A Java startup applet template is also. . .

DETDESC:

DETD(97)

Presentation . . . is provided as an example in accordance with a preferred embodiment. Developing a specific application presentation engine means extending and **customizing** the basic presentation engine framework template. A Presentation Engine (PE) is itself a system with two components: (1) a UI. . .

DETDESC:

DETD(119)

FIG. . . . at 1810. In addition, user input is needed to modify the Comm Layer 1840, the mediator 1860, ~~add-to-the-Custom~~ Comm Mechanism 1870 and the **Custom** Input Mechanism 1880. All of the aforementioned elements communicate with each other through Mediator 1860. The mediator 1860 communicates, in. . .

DETD(DESC:

DETD(126)

Enterprise . . . program or programs for accessing the information on the server. The complexity of these environments makes it difficult to create, **configure**, deploy, and administer software applications. The advent of Web technologies (browsers, the Java language, HTTP) has enabled enterprises to create. . .

DETD(DESC:

DETD(146)

A **customizable** Access Layer, installed on the server, enables centralized control of access to client programs.

DETD(DESC:

DETD(153)

An . . . also provides a template for creating a startup applet that enables users to start applications from a browser. Chapter 3, "**Configuring** and Deploying ICE-T Applications" describes these tasks and tools. ((

DETD(DESC:

DETD(465)

```
### Environment Configuration ###
#####- #
#Change the following lines to indicate the location of your compiler.
COMPILER.sub.-- BIN=
COMPILER.sub.-- LIB=
#
#.
```

DETD(DESC:

DETD(467)

```
#####- ### PE ###
### (Configurable Macros)###
#####- # Change the following macro to add your
java files
PE.sub.-- SOURCE5.java= .backslash.
    myGui.java .backslash.
    pe.sub.-- . . .
```

DETD(DESC:

DETD(469)

```

### Server #####
### (Configurable Macro) ###
#####
# change the following macro to add .cc (C Plus Plus) files
SERVER.sub.-- SOURCES.cc=.backslash.
#end
#. . .

```

DETD(493)

Configuring and Deploying ICE-T Applications

DETD(504)

Acts . . . can modify the e files used by the Access Layer and then use the supplied makefile (Access.mk) to build a **customized** Access program for use with ICE-T server applications.

DETD(513)

3. **Customizing** (optional) and installing the Access Layer

DETD(515)

5. **Configuring** application management files

DETD(563)

6. Provide the name of the access program to use for this application. The default value is "Access." You may have **customized** the default access file and given it another name. If so, provide the name as a value to the Access. . . .

DETD(580)

As part of the process of determining access and downloading Presentation Engines, the Access Layer relies on an application configuration file to provide information about the location and names of program components. ICE-T's Access Layer installation script generates the application configuration file automatically. That configuration is the basis for generating an HTML wrapper file in which to download the Presentation Engine. You can accept the defaults and let your application use the generated HTML wrapper, or you can customize the application configuration file so that it generates a customized HTML file to hold the Presentation Engine. See "Configuring Applications" for more information. //

DETD(590)

If . . . you will need to change the Access Layer's default access properties file to specify the cgi-bin and httpd-docs locations. (See "Customizing the Access Layer"). If you have created the links, run ice-httpd-setup without arguments.

DETD(590)

DETD(609)

Customizing the Access Layer

DETD(DESC:

DETD(634)

5. After you **configure** the Access Layer, run `ice- install -access`. By default, Access .ink creates a an executable named **CustomAccess**. If you have changed the default name of the access program, use the `--source` option to `ice-install-access` and specify the name. . .

DETD(DESC:

DETD(659)

Creates an application **configuration** file (`appConfigFile`), or installs a specified one

DETD(DESC:

DETD(679)

Configuring Applications

DETD(DESC:

DETD(680)

The ICE-T application installation script (`ice-app- install`) generates a default application **configuration** file (`appConfigFile`). When a user starts an application by launching a startup applet, the Access Layer uses the Application Manager to look up values for server and client program locations and names in `appConfigFile`. Using the **configuration** file, the Application Manager generates an HTML wrapper for presenting Presentation Engines as applets in a Web browser execution environment..

DETD(DESC:

DETD(683)

Or, create your own application **configuration** file by modifying the automatically generated `appConfigFile`.

DETD(DESC:

DETD(685)

To **customize** the `appConfigFile` generated by `ice-app- install`:

DETD(DESC:

DETD(686)

1. Open the generated **configuration** file (`<appName>.appconfl`) in an editor.

DETD(DESC:

DETD(696)

The . . . Access Layer. You can specify another application to return by modifying `default.sub.-- tappmgr.sub.-- properties. cc.default.sub.-- appmgr.sub.-- properties.cc` is described in "**Customizing the Access Layer**". ##SPC1##

CLAIMS:

CLMS(1)

Having . . .
computer code segment resident on a server computer node coupled to the
client computer node;
(c) an execution framework code segment **configured** to couple the
client code segment and the server code segment to event driven message
transfer between the client computer. . . including:
(1) a user interface resident on the client computer node;
the client computer code segment, including:
(a) a user **configurable** user interface adaptor coupled between
the user interface and the execution framework code segment.

CLAIMS:

CLMS(8)

8. . . . system and a client computer system coupled by a network,
comprising the step of:
storing an execution framework code segment **configured** to couple a
client code segment resident on the client computer system and a server
code segment resident on the. . . segment, including:
(1) a user interface resident on the client computer system; the client
computer code segment, including:
(a) a user **configurable** user interface adaptor coupled between the
user interface and the execution framework code segment.

CLAIMS:

CLMS(15)

15. . . .
responding to a request from a client computer system to a server
computer system; and
(b) an execution framework code segment **configured** to couple the
server computer and the client computer via the network, comprising:
(1) a plurality of client computer code. . . including:
(a1) a user interface resident on the client computer node, the client
computer code segment, including:
(a2) a user **configurable** user interface adaptor coupled between
the user interface and the execution framework code segment.

=> s Sun Microsystem? /asn

```
          2355 SUN/ASN
          1343 MICROSYSTEM?/ASN
L9          951 SUN MICROSYSTEM? /ASN
            ((SUN(W)MICROSYSTEM?)/ASN)
```

=> s java

```
L10          234 JAVA
```

=> s l9 and l10

```
L11          27 L9 AND L10
```

=> s hotjava

```
L12          14 HOTJAVA
```

=> s l11 and l12

```
L13          9 L11 AND L12
```

=> d 1-9

1. 5,802,530, Sep. 1, 1998, Web document based graphical user interface; Arthur A. Van Hoff, 707/513; 345/335; 395/200.33, 682 [IMAGE AVAILABLE]

2. 5,794,049, Aug. 11, 1998, Computer system and method for executing architecture specific code with reduced run-time memory space requirements; Timothy G. Lindholm, 395/706, 704, 709 [IMAGE AVAILABLE]

3. 5,790,855, Aug. 4, 1998, System, method and article of manufacture for type checking appropriateness of port connection and variable type matching in connection with multiport object-oriented components; Antony Azio Faustini, 395/701; 345/348 [IMAGE AVAILABLE]

4. 5,765,157, Jun. 9, 1998, Computer system and method for executing threads of execution with reduced run-time memory space requirements; Timothy G. Lindholm, et al., 707/101; 341/51, 79; 707/205, 206; 711/113, 160 [IMAGE AVAILABLE]

5. 5,761,513, Jun. 2, 1998, System and method for exception handling in dynamically linked programs; Frank Yellin, et al., 395/705, 704 [IMAGE AVAILABLE]

6. 5,761,421, Jun. 2, 1998, System and method for secure peer-to-peer communication between downloaded programs; Arthur A. van Hoff, et al., 395/200.53; 340/825.5; 395/200.58 [IMAGE AVAILABLE]

7. 5,754,857, May 19, 1998, Distributed asynchronous workflow on the net; Steven D. Gadol, 395/680, 684 [IMAGE AVAILABLE]

8. 5,736,984, Apr. 7, 1998, Method and system for embedded feedback message and graphical processing element; Herb Jellinek, et al., 345/338, 335 [IMAGE AVAILABLE]

9. 5,727,147, Mar. 10, 1998, System and method for resolving symbolic